Additive First-Order Queries

Gerald Berger
TU Wien, Austria

Martin Otto
TU Darmstadt, Germany

Andreas Pieris
University of Edinburgh, Scotland

Dimitri Surinx
Hasselt University, Belgium

Jan Van den Bussche
Hasselt University, Belgium

Abstract

A database query q is called additive if $q(A \cup B) = q(A) \cup q(B)$ for domain-disjoint input databases A and B. Additivity allows the computation of the query result to be parallelised over the connected components of the input database. We define the "connected formulas" as a syntactic fragment of first-order logic, and show that a first-order query is additive if and only if it expressible by a connected formula. This characterisation specializes to the guarded fragment of first-order logic. We also show that additivity is decidable for formulas of the guarded fragment, establish the computational complexity, and do the same for positive-existential formulas. Our results hold when restricting attention to finite structures, as is common in database theory, but also hold in the unrestricted setting.

2012 ACM Subject Classification Information systems → Query languages

Keywords and phrases Expressive power

Digital Object Identifier 10.4230/LIPIcs.ICDT.2019.19

1 Introduction

Motivated by cloud computing and big data, in the past few years, there has been interest in logical characterisations of queries that can be answered using parallelism [11]. An attractive class of queries that was identified is formed by those that "distribute over components" [3]. These queries can be faithfully answered by the following three-step strategy: first, partition the input database in any way that does not split connected components; second, evaluate the query separately on each part, thus allowing some degree of parallelism; third, collect the final result by taking the union of all partial results. Equivalently, over finite databases, a query q for which this works correctly is *additive*, meaning that $q(A \cup B) = q(A) \cup q(B)$ for any two domain-disjoint databases A and B. An added bonus is that the two partial results $Q(A)$ and $Q(B)$ are disjoint. Apart from the relevance to distributed computing, additivity is also a useful notion in the analysis of expressive power of query languages [20, 21].

Additivity is a semantic property that is undecidable for queries given, say, as Datalog programs, or as first-order logic formulas. It is therefore desirable to match additivity to a syntactic restriction in the query language. This was done successfully in the setting of Datalog programs, where it is a natural idea to restrict rule bodies so that they must be connected. It is then relatively straightforward to show that a Datalog query is additive if and only if it can be computed by a Datalog program with connected rule bodies. This program was successfully carried out for a range of Datalog variants, such as Datalog extended

with stratified or well-founded negation, or value invention [3, 4], as well as for unions of conjunctive queries, and ontology-mediated querying with conjunctive queries over linear, guarded, or sticky tuple-generating dependencies [6].

The case of first-order queries (equivalently, relational algebra queries), however, has remained open until now. In this paper we define a syntactical notion of connectedness for first-order logic formulas, as well as for relational algebra expressions. In a connected relational algebra expression, equijoins are allowed, but cartesian products are not. Connected first-order logic formulas are defined similarly. Queries expressible by connected formulas are always additive. We show that the converse holds as well: if a first-order formula φ expresses an additive query, then φ is equivalent to a connected formula. A similar result then follows for relational algebra expressions.

Results of this kind are colloquially known as expressive completeness results, characterisation theorems, or preservation theorems [2, 18]. For example, modal characterisation theorems link bisimulation-invariant first-order formulas (in one free variable) to formulas in modal logics [23, 17, 14, 15, 16]. Indeed, the proof of our result starts from ideas that were developed to prove modal characterisation theorems over finite structures as well as over all structures. As a consequence, also our result holds over finite structure as well as over all structures.

In a second part of the paper, we consider the guarded fragment (GF) of first-order logic [5]. GF was originally introduced as a first-order generalisation of modal logic that preserves good properties such as the tree model property, the finite model property, and a decidable satisfiability problem. Satisfiability for GF is 2ExpTime-complete in general and ExpTime-complete for bounded arity [9].

When restricting attention to queries returning guarded tuples, the guarded fragment corresponds to the semijoin algebra [12]. Such queries are always additive. We complement this picture and show that a guarded formula φ expresses an additive query if and only if φ is equivalent to a connected guarded formula. We will see that additivity is decidable for GF as well, by a reduction to satisfiability. While our reduction preserves bounded arity, it is unfortunately exponential, so we only obtain a 2ExpTime upper bound, even for bounded arity. By a very simple reduction from satisfiability, additivity for GF is 2ExpTime-complete in general and ExpTime-hard for bounded arity.

Finally, we will consider positive-existential (PE) first-order formulas. Such formulas have the same expressive power as unions of conjunctive queries (UCQ) [1]. In earlier work [6], additivity for UCQs was shown to be NP-complete. In this paper we complement this result and show that additivity for PE formulas is Π_2^P-complete. The pattern that seems to emerge here is that additivity has the same complexity as containment.

2 Preliminaries

From the outset, we assume a supply of *relation names*, where each relation name has an associated arity (a natural number). We use R/k to denote that the relation name R has arity k. In this paper we do not consider nullary relation names, as their presence corrupts the intuitive notion of domain-disjointness which will play an important role in this work (see Section 3).

We further assume an infinite domain **dom** of atomic data elements called *constants*. A *fact* is of the form $R(a_1, \ldots, a_k)$ where $a_1, \ldots, a_k$ are constants and R/k is a relation name. We call R the *predicate* of the fact.

A *database schema* (or schema for short) is a finite set of relation names. An *instance* of a schema $\mathbf{S}$ is a nonempty set of facts with predicates from $\mathbf{S}$. The set of all constants appearing in an instance $\mathcal{A}$ is called the *active domain* of $\mathcal{A}$ and denoted by $adom(\mathcal{A})$.

▶ Remark (Finite vs unrestricted instances). In database theory it is common and natural to restrict attention to finite instances. Our results will hold in restriction to finite instances, as well as in the unrestricted setting where we allow all instances.

Let $\mathbf{S}$ be a schema, and let k be a natural number. A *k-ary query over* $\mathbf{S}$ is a function that maps each instance $\mathcal{A}$ of $\mathbf{S}$ to a k-ary relation on $adom(\mathcal{A})$. Note that a nullary query ($k = 0$) has only two possible results, $\{()\}$ and $\emptyset$, which can be interpreted as the boolean values true and false respectively. Thus nullary queries capture the notion of boolean or yes/no query.

A standard language for expressing queries is the relational algebra [22]. To fix notation, we define it formally [13]. Note that we only consider equality selections and equijoins; the extension of our work to selection and join predicates involving constants, order, or arithmetic, is an interesting direction for further research.

▶ **Definition 1.** *The expressions of the relational algebra (RA) over a schema* $\mathbf{S}$ *are generated as follows:*

- *Each relation name* $R \in \mathbf{S}$ *is a relational algebra expression. Its arity comes from* $\mathbf{S}$.
- *If* $E_1, E_2 \in RA$ *have arity* n*, then also* $E_1 \cup E_2$ *(union) and* $E_1 - E_2$ *(difference) belong to RA and are of arity* n.
- *If* $E \in RA$ *has arity* n *and* $i_1, \ldots, i_k \in \{1, \ldots, n\}$*, then* $\pi_{i_1,\ldots,i_k}(E)$ *(projection) belongs to RA and is of arity* k.
- *If* $E \in RA$ *has arity* n *and* $i, j \in \{1, \ldots, n\}$*, then* $\sigma_{i=j}(E)$ *(selection) belongs to RA and is of arity* n.
- *If* $E_1, E_2 \in RA$ *have arities* n *and* m*, respectively, then* $E_1 \times E_2$ *(cartesian product) belongs to RA and is of arity* $n + m$.

The semantics is well known; we recall it for the sake of completeness. Let $\mathcal{A}$ *be an instance of* $\mathbf{S}$.

- *If* E *is a relation name* R *then* $E(\mathcal{A}) := \{\bar{a} \mid R(\bar{a}) \in \mathcal{A}\}$.
- *If* E *is* $E_1 \cup E_2$*, or* $E_1 - E_2$*, or* $E_1 \times E_2$*, then* $E(\mathcal{A}) := E_1(\mathcal{A}) \cup E_2(\mathcal{A})$*, or* $E_1(\mathcal{A}) - E_2(\mathcal{A})$*, or* $E_1(\mathcal{A}) \times E_2(\mathcal{A})$*, respectively.*
- *If* E *is* $\pi_{i_1,\ldots,i_k}(E_1)$ *then* $E(\mathcal{A}) := \{(a_{i_1}, \ldots, a_{i_k}) \mid \bar{a} \in E_1(\mathcal{A})\}$.
- *If* E *is* $\sigma_{i=j}(E_1)$ *then* $E(\mathcal{A}) := \{\bar{a} \in E_1(\mathcal{A}) \mid a_i = a_j\}$.

The relational algebra is equivalent in expressive power to the language of first-order logic (FO) [22, 1]. To match RA as formalised above, we use FO with equality but without constants. An FO formula φ with free variables $x_1, \ldots, x_k$ expresses the k-ary query that maps an instance $\mathcal{A}$ to the set of all k-tuples $(\alpha(x_1), \ldots, \alpha(x_k))$, where α is a valuation from $\{x_1, \ldots, x_k\}$ to $adom(\mathcal{A})$ such that $\mathcal{A}, \alpha \models \varphi$. Here, $\mathcal{A}$ is viewed as a structure with domain $adom(\mathcal{A})$, so we employ the active-domain semantics for first-order logic. We denote the resulting relation by $\varphi(\mathcal{A})$. In particular, sentences (formulas without free variables) express nullary queries.

Notation.

- To indicate an ordering of the free variables of φ, we will use the notation $\varphi(x_1, \ldots, x_k)$.
- We will use FO to denote the class of queries expressible in FO.

3 Additive queries

Let q be a query over schema $\mathbf{S}$. We call q *additive* if $q(\mathcal{A} \cup \mathcal{B}) = q(\mathcal{A}) \cup q(\mathcal{B})$ for any two $\mathbf{S}$-instances $\mathcal{A}$ and $\mathcal{B}$ that are *domain-disjoint*, meaning that $adom(\mathcal{A})$ is disjoint from $adom(\mathcal{B})$. Note that, formally, this definition applies equally well to boolean (nullary)

queries. Interpreting $\{()\}$ as true and $\emptyset$ as false, as we will always do, the condition $q(\mathcal{A} \cup \mathcal{B}) = q(\mathcal{A}) \cup q(\mathcal{B})$ can be equivalently read as "$q(\mathcal{A} \cup \mathcal{B})$ is true if and only if $q(\mathcal{A})$ is true or $q(\mathcal{B})$ is true".

Notation. When we write $\mathcal{A} = \mathcal{B} + \mathcal{C}$ we will mean that $\mathcal{A} = \mathcal{B} \cup \mathcal{C}$ and $\mathcal{B}$ and $\mathcal{C}$ are domain-disjoint.

▶ **Example 2.** Consider queries about two binary relations R and T, expressed in RA.

1. The selection $\sigma_{1=2}(R)$ is clearly additive.
2. The join $\pi_{1,2,4}\sigma_{2=3}(R \times T)$ is additive.
3. The union $R \cup T$ and the difference $R - T$ are additive.

In order to give examples of queries that are not additive, it is useful to introduce the following definitions and lemma.

▶ **Definition 3** ([3]). *A* strict component *of an instance $\mathcal{A}$ is an instance $\mathcal{C}$ such that $\mathcal{A} = \mathcal{C} + \mathcal{B}$ for some instance $\mathcal{B}$, and $\mathcal{C}$ is minimal with this property. (Recall that instances are nonempty.) We now say that an instance $\mathcal{C}$ is a* component *of an instance $\mathcal{A}$, if $\mathcal{C}$ is a strict component of $\mathcal{A}$, or if $\mathcal{C}$ equals $\mathcal{A}$ and $\mathcal{A}$ has no strict components.*

▶ **Definition 4.** *Let $\mathcal{A}$ be an instance and let t be a tuple of elements of $adom(\mathcal{A})$. We call t* mixed *with respect to $\mathcal{A}$, if t contains elements from (at least) two different components of $\mathcal{A}$.*

▶ **Lemma 5.** *Let q be an additive query and let $\mathcal{A}$ be an instance. Then $q(\mathcal{A})$ does not contain any mixed tuples (with respect to $\mathcal{A}$).*

Proof. Suppose $t \in q(\mathcal{A})$ contains elements from two distinct components $\mathcal{B}$ and $\mathcal{C}$ (and possibly more). Write $\mathcal{A} = \mathcal{B} + \mathcal{C} + \mathcal{D}$. Then $t \in q(\mathcal{B}) \cup q(\mathcal{C}) \cup q(\mathcal{D})$. However, t cannot be in $q(\mathcal{B})$ since $q(\mathcal{B})$ is a relation on $adom(\mathcal{B})$ and t contains elements from $adom(\mathcal{C})$. For similar reasons, t can neither be in $q(\mathcal{C})$ nor in $q(\mathcal{D})$. We have arrived at a contradiction. ◀

We can now follow up Example 2 with some negative examples.

▶ **Example 6.**

1. Let E be the cartesian product $R \times T$. Then E is not additive. Indeed, consider $\mathcal{A} = \{R(a, a), T(b, b)\}$. Then $(a, a, b, b) \in E(\mathcal{A})$ is a mixed tuple, contradicting Lemma 5.
2. Let $\varphi(x, y, z)$ be the FO query $R(x, y) \vee T(y, z)$. Then φ is not additive. Indeed, consider $\mathcal{A} = \{R(a, b), T(c, d)\}$. Then $(a, b, c) \in \varphi(\mathcal{A})$ is a mixed tuple, again contradicting Lemma 5.
3. Let φ be the boolean FO query $\neg\exists z\, R(z, z)$. Then φ is not additive. Indeed, consider $\mathcal{A} = \{R(a, b)\}$ and $\mathcal{B} = \{R(c, c)\}$. Then $\varphi(\mathcal{A})$ is true, but $\varphi(\mathcal{A} \cup \mathcal{B})$ is false. Note that φ does not return mixed tuples, yet is not additive.

3.1 Queries that distribute over components

A notion closely related to additivity is that of *distributing over components* [3]. A query q is said to distribute over components if $q(\mathcal{A})$ equals the union of $q(\mathcal{C})$ over all components $\mathcal{C}$ of $\mathcal{A}$.

Let us denote the class of additive queries by ADD and the class of queries distributing over components by DIST. Clearly, DIST implies ADD. Over instances that have only finitely many components, the converse implication is quite clear as well. Over infinite instances, the converse implication can still be shown quite simply, but only for non-boolean queries.

We summarise the situation as follows. Let ADD* be the class of additive queries that are not nullary, and similarly for DIST*.

▶ **Proposition 7.** *In restriction to finite instances,* ADD = DIST. *In the unrestricted setting,* $\text{ADD}^* = \text{DIST}^*$.

Proof. The only part that is not immediately clear is that ADD implies DIST for non-boolean queries in the unrestricted setting. To show this, let q be an additive non-boolean query, and let $\mathcal{A}$ be an instance. We need to verify that $q(\mathcal{A})$ equals the union of $q(\mathcal{C})$ over all components $\mathcal{C}$ of $\mathcal{A}$. For the inclusion from right to left, take a component $\mathcal{C}$ and write $\mathcal{A} = \mathcal{C} + \mathcal{B}$. Then $q(\mathcal{C}) \subseteq q(\mathcal{C}) \cup q(\mathcal{B}) = q(\mathcal{A})$.

For the inclusion from left to right, let $t \in q(\mathcal{A})$. By Lemma 5, t consists of elements from a single component $\mathcal{C}$ of $\mathcal{A}$. Moreover, since q is non-boolean, t is nonempty. Write $\mathcal{A} = \mathcal{C} + \mathcal{B}$. Thus $t \in q(\mathcal{C}) \cup q(\mathcal{B})$. However, t cannot be in $q(\mathcal{B})$ since $\mathcal{B}$ is domain-disjoint from $\mathcal{C}$. Hence $t \in q(\mathcal{C})$ as desired. ◀

Actually, over infinite instances, ADD does not imply DIST for boolean queries. Indeed, the boolean query "does the instance have infinitely many components?" is in ADD but not in DIST. Within FO, however, the equivalence between ADD and DIST does hold. This will follow from our result on additive boolean FO queries (Proposition 14).

4 The first-order case

In this section we introduce the connected formulas as a syntactical restriction on FO formulas. Our main result will be that a query expressible in FO is additive if and only if it is expressible by a connected FO formula. A similar result will then follow for the relational algebra. We will conclude the section with a very simple reduction from satisfiability to additivity.

4.1 Connected FO

Connected FO formulas are generated according to the following syntax rules:

- Every atomic formula is connected.
- If φ is connected then $\exists y\, \varphi$, for any variable y, is also connected.
- If φ and ψ are connected and they share at least one free variable, then $\varphi \wedge \psi$ is also connected.
- If φ and ψ are connected and have exactly the same free variables (possibly none), then $\varphi \vee \psi$ is also connected.
- If φ and ψ are connected, ψ has at least one free variable, and φ has all the free variables of ψ, then $\varphi \wedge \neg\psi$ is also connected.

▶ **Example 8.** Revisiting Examples 2 and 6, we see that the positive examples are expressible by connected formulas: the selection example by $R(x, y) \wedge x = y$; the equijoin by $R(x, y) \wedge T(y, z)$; the union and difference by $R(x, y) \vee T(x, y)$ and $R(x, y) \wedge \neg T(x, y)$, respectively.

We also see that the negative examples are not connected formulas: the formula $R(x, y) \wedge T(u, v)$ for cartesian product violates the conjunction rule; the formula $R(x, y) \vee T(y, z)$ violates the disjunction rule; and the formula $\neg\exists z\, R(z, z)$ violates the negation rule.

It is readily verified that connected FO queries are always additive. Our main result in this section is that the converse holds as well. Let CFO denote the queries expressible by a connected FO formula. We establish:

▶ **Theorem 9.** $\text{FO} \cap \text{ADD} = \text{CFO}$.

4.2 Non-boolean queries

Our theorem is proven separately for formulas with free variables, and for sentences. In this subsection we handle the case with free variables.

We need to recall the basic notions concerning the locality of first-order logic [8]. The *Gaifman graph* of an instance $\mathcal{A}$, denoted by $G(\mathcal{A})$, is the undirected graph with $adom(\mathcal{A})$ as the set of nodes; there is an edge between distinct nodes a and b if a and b occur together in some fact in $\mathcal{A}$. The distance between a and b in $G(\mathcal{A})$ is denoted by $d^{\mathcal{A}}(a,b)$, or simply $d(a,b)$ if $\mathcal{A}$ is understood. For any natural number ℓ, the set $\{b \in adom(\mathcal{A}) \mid d(a,b) \leq \ell\}$ is denoted by $B^{\mathcal{A}}(a,\ell)$ or $B(a,\ell)$ if $\mathcal{A}$ is understood (B for "ball"). The notation $B(\bar{a},\ell)$, for a tuple $\bar{a} = a_1,\ldots,a_k$, denotes the union of $B(a_i,\ell)$ for $i = 1,\ldots,k$. The restriction of $\mathcal{A}$ to $B(\bar{a},\ell)$ is denoted by $N^{\mathcal{A}}(\bar{a},\ell)$ (N for "neighbourhood"). A k-ary formula $\varphi(\bar{x})$ is called *ℓ-local* if for every instance $\mathcal{A}$ and every k-tuple $\bar{a}$ over $adom(\mathcal{A})$, we have

$$\bar{a} \in \varphi(\mathcal{A}) \quad \Leftrightarrow \quad \bar{a} \in \varphi(N^{\mathcal{A}}(\bar{a},\ell)).$$

We say that a formula is local if it is ℓ-local for some ℓ.

We are now ready to state an important lemma. In modal characterisation theorems, one considers unary FO formulas over unary and binary relations that are invariant under bisimulation [23]. Invariance under bisimulation implies invariance under disjoint sums [15]. Additivity is the natural generalisation of invariance under disjoint sums to higher-arity queries. In earlier work, one of us showed, by a detailed Ehrenfeucht-Fraïssé game argument, that unary FO formulas, invariant under disjoint sums, are always local [16, Lemma 3.5]. We have carefully verified that the exact same argument also works for formulas about higher-arity relations and with multiple free variables. We thus obtain:

▶ **Lemma 10.** *Let φ be an additive FO formula and let q be its quantifier rank. Then φ is 2^q-local.*

By the above lemma, all subformulas of $\varphi(\bar{x})$ may be assumed to talk about elements y that belong to $B(\bar{x},2^q)$. Moreover, by Lemma 5 we already know that the values of the free variables $\bar{x}$ must come from the same component. This is not quite enough, however, to express $y \in B(\bar{x},2^q)$ by a connected FO formula; for that we need a finite upper bound on the diameter of the tuple of (valuations of) free variables that holds uniformly over all instances. This is provided by the following:

▶ **Proposition 11.** *Let φ be an additive FO formula of quantifier rank q. Assume $(a_1,\ldots,a_n)$ is in $\varphi(\mathcal{A})$. Then $d^{\mathcal{A}}(a_i,a_j) \leq (n-1)2^q$ for all $i,j \in \{1,\ldots,n\}$.*

Proof. Let $\bar{a} = a_1,\ldots,a_n$ and let $\ell = 2^q$. Suppose to the contrary that $d(a_i,a_j) > (n-1)\ell$. Then there exists a partition $\{I,J\}$ of $\{1,\ldots,n\}$ such that $d(\bar{a}|_I,\bar{a}|_J) > \ell$. Here, $d(\bar{a}|_I,\bar{a}|_J)$ naturally stands for the minimum of $d(a_i,a_j)$ for $i \in I$ and $j \in J$.

Let us make a domain-disjoint copy $f(\mathcal{A})$ of $\mathcal{A}$ by using a bijection $f : adom(\mathcal{A}) \to B$ for some set B disjoint from $adom(\mathcal{A})$. Take q additional domain-disjoint copies of $\mathcal{A}$, which we denote by $q \cdot \mathcal{A}$. Define the tuple $\bar{b} = b_1,\ldots,b_n$ by

$$b_i = \begin{cases} a_i & \text{if } i \in I, \\ f(a_i) & \text{if } i \in J. \end{cases}$$

Consider the instance $\mathcal{C} = \mathcal{A} + f(\mathcal{A}) + q \cdot \mathcal{A}$. Since $\bar{a} \in \varphi(\mathcal{A})$ and φ is additive, also $\bar{a} \in \varphi(\mathcal{C})$. Using an Ehrenfeucht-Fraïssé game argument, inspired on the original proof of Lemma 10, we obtain that $\bar{b} \in \varphi(\mathcal{C})$. This contradicts the additivity of φ, since $\bar{b}$ is a mixed tuple with respect to $\mathcal{C}$ (Lemma 5). ◀

Our final lemma is the following:

▶ **Lemma 12.** *Let $\bar{x}$ be a nonempty list of variables. Let φ be a formula, with free variables among $\bar{x}$, so that each quantifier in φ is of the form $\exists y \in B(\bar{x}', \ell)$ with $y \notin \bar{x}$, and $\bar{x}'$ a nonempty subtuple of $\bar{x}$. Let δ be a connected formula with exactly $\bar{x}$ as free variables. Then $\varphi \wedge \delta$ can be rewritten as a connected formula.*

Proof. It is readily verified that predicates of the form $d(x, y) \leq m$ can be expressed by connected formulas. The lemma can be proven by a straightforward structural induction. ◀

▶ **Example 13.** Let us illustrate Lemma 12 with φ being $\exists y \in B(x_1, x_2, 3)\, \neg S(y)$, and δ being $R(x_1, x_2)$. The formula $\varphi \wedge \delta$ can be rewritten into the connected formula

$$\exists y\big((\neg S(y) \wedge d(x_1, y) \leq 3 \wedge R(x_1, x_2)) \vee (\neg S(y) \wedge d(x_2, y) \leq 3 \wedge R(x_1, x_2))\big).$$

We now have all ingredients to prove that every additive FO formula φ with at least one free variable can be rewritten as a connected formula. Let $\bar{x} = x_1, \ldots, x_n$ be the list of free variables of φ. By Lemma 10, we may assume that each quantifier in φ is of the form $\exists y \in B(\bar{x}, \ell)$. We may always assume that $y \notin \bar{x}$ simply by choosing another bound variable. Furthermore, by Proposition 11, φ is equivalent to $\varphi \wedge \delta$, where δ is the conjunction of $d(x_1, x_i) \leq (n-1)\ell$ for $i = 2, \ldots, n$. (If $n = 1$, set δ to $x_1 = x_1$.) Hence, by Lemma 12, φ is equivalent to a connected formula as desired.

4.3 Boolean queries

The argument in the previous subsection relies on the presence of at least one free variable to connect everything inside the formula. So, to prove Theorem 9 for sentences, we need a separate argument. Interestingly, this argument will then only work for sentences and not with free variables. However, in the next section, we will be able to adapt the argument at least to guarded formulas, with or without free variables.

Recall that a *simple local sentence* is a sentence of the form $\exists x\, \varphi$ where φ is local [15]. By Lemma 12, such sentences can be rewritten in connected form (use $x = x$ for δ). Since a disjunction of connected sentences is again connected, the following proposition proves all we need.

▶ **Proposition 14.** *Every additive FO sentence can be rewritten as a finite disjunction of simple local sentences.*

Proof. Let φ be an additive sentence. The formula φ^* that is $(x = x) \wedge \varphi$ is *invariant under disjoint copies*, by which we mean the following. For an instance $\mathcal{A}$, let $q \cdot \mathcal{A}$ denote an instance consisting of q domain-disjoint copies of $\mathcal{A}$. Then, for any $a \in adom(\mathcal{A})$ and any q, we have $a \in \varphi^*(\mathcal{A})$ iff $a \in \varphi^*(\mathcal{A} + q \cdot \mathcal{A})$. Over unary and binary relations, it was already proven that any formula in one free variable, invariant under disjoint copies, is equivalent to a boolean combination of simple local sentences and local formulas [15, Proposition 19]. That proof goes through verbatim for higher-arity relations as well. It follows that $\varphi \equiv \exists x\, \varphi^*$ is equivalent to a boolean combination of simple local sentences.

So we now assume that φ is a boolean combination, in disjunctive normal form, over a finite set Σ of simple local sentences. We may assume that each clause is *complete*, meaning that it either contains σ or $\neg\sigma$ for every $\sigma \in \Sigma$. We may also assume that each clause is satisfiable. We will identify φ with the set of its clauses.

The plan of the proof is to first get rid of negations, then of conjunctions, leaving a disjunction as desired. For the first step, let Γ denote the set of all satisfiable complete clauses over Σ. For two clauses γ_1 and γ_2 in Γ, we write $\gamma_1 \leq \gamma_2$ if every $\sigma \in \Sigma$ that occurs positively in γ_1 also occurs positively in γ_2. We claim the following *monotonicity property:*

Let $\gamma_1 \in \varphi$ and let $\gamma_2 \in \Gamma$ such that $\gamma_1 \leq \gamma_2$. Then also $\gamma_2 \in \varphi$.

In proof, let $\mathcal{A}$ and $\mathcal{B}$ be domain-disjoint instances such that $\mathcal{A} \models \gamma_1$ and $\mathcal{B} \models \gamma_2$. Since $\mathcal{A} \models \varphi$, by additivity also $\mathcal{A} + \mathcal{B} \models \varphi$, so $\mathcal{A} + \mathcal{B} \models \gamma$ for some $\gamma \in \varphi$. We verify that $\gamma = \gamma_2$. Consider any $\sigma \in \Sigma$.

- If σ occurs positively in γ_2, then $\mathcal{B} \models \sigma$, so also $\mathcal{A} + \mathcal{B} \models \sigma$. Hence, σ cannot occur negated in γ, i.e., σ occurs positively also in γ.
- If σ occurs negated in γ_2, then also in γ_1, so neither $\mathcal{A} \models \sigma$ nor $\mathcal{B} \models \sigma$ holds, and thus $\mathcal{A} + \mathcal{B} \models \neg\sigma$. Hence, σ cannot occur positively in γ, i.e., σ occurs negated also in γ.

By the monotonicity property, we can simplify φ so that it is now a disjunction of conjunctions, each conjunction over some finite subset of Σ. We will naturally view φ as a set of subsets of Σ. We may also assume that φ is simplified further so that every $\gamma \in \varphi$ is *minimal*, meaning that replacing γ by a strict subset $\gamma' \subsetneq \gamma$ would yield a sentence not equivalent to φ.

Now assume, for the sake of contradiction, that some $\gamma \in \varphi$ has at least two elements. Then we can split $\gamma = \gamma_1 \cup \gamma_2$ in two disjoint nonempty subsets. By the minimality assumption, there exist instances $\mathcal{A}$ and $\mathcal{B}$ such that $\mathcal{A} \models \gamma_1 \wedge \neg\varphi$, and $\mathcal{B} \models \gamma_2 \wedge \neg\varphi$. (Here, we view each γ_i as the conjunction of its elements.) By additivity, $\mathcal{A} + \mathcal{B} \models \neg\varphi$. However, clearly $\mathcal{A} + \mathcal{B} \models \gamma$, whence $\mathcal{A} + \mathcal{B} \models \varphi$, a contradiction. ◀

4.4 Some consequences

Connected RA. We can give an analogue to Theorem 9 for RA instead of FO. Call an RA expression *connected* if every cartesian product subexpression of the form $e_1 \times e_2$ occurs in the context of a selection subexpression of the form $\sigma_{i=j}(e_1 \times e_2)$, with $i \in \{1, \ldots, n\}$ and $j \in \{n+1, \ldots, i+m\}$, where n and m are the arities of e_1 and e_2 respectively. In other words, in connected RA, pure cartesian products are disallowed, but equijoins are still allowed. Denote the queries expressible by connected RA expressions by CRA. It is clear that CRA queries are always additive. Again we have the converse:

▶ **Corollary 15.** $\mathrm{RA} \cap \mathrm{ADD} = \mathrm{CRA}$.

Indeed, it is well known that RA can be translated into FO, so, by Theorem 9, all we need to show is that CFO can be translated back into CRA. The translation from FO to RA given by Ullman [22] can be easily adapted to this effect.

Finite vs infinite. Theorem 9 holds in restriction to finite instances, as well as over all instances, but the two settings still need to be kept separate. CFO formulas are always additive, over all instances. This, however, should not lull the reader into believing that an FO formula that is additive over finite instances is also additive over all instances. Indeed, our results only show that such a formula is equivalent to a CFO formula over finite instances.

An example of an FO sentence over a binary relation R that is additive over finite instances but not in general, expresses that R consists of two total orders, over two disjoint sets, each order without a maximum.

4.5 Reduction from satisfiability

It is expected that additivity of FO formulas is an undecidable property. There is actually an extremely simple reduction from unsatisfiability. The proof of the correctness of this reduction, while short, is not entirely trivial and given in the Appendix.

▶ **Proposition 16.** *Let φ be an FO sentence over schema $\mathbf{S}$ and let S and T be two unary relation names not in $\mathbf{S}$. Then φ is unsatisfiable if and only if $\varphi \wedge \exists x\, S(x) \wedge \exists y\, T(y)$ is additive.*

Over all instances, the additive FO formulas are recursively enumerable; this actually follows from our theorem, but can also be seen by a direct reduction from additivity to unsatisfiability. We will see this reduction later when showing decidability of validity for guarded formulas. Over finite instances, the additive FO formulas are clearly co-r.e.

5 The guarded case

In this section we specialise Theorem 9 to the guarded fragment. We also show that additivity for guarded formulas is decidable and 2ExpTime-complete.

5.1 Guarded and connected formulas

We recall the syntax of the guarded fragment [5], which we denote by GF.

- Every atomic formula is guarded.
- If φ and ψ are guarded, then so are $\varphi \wedge \psi$, $\varphi \vee \psi$, and $\neg\varphi$.
- If α is an atomic formula, ψ is a guarded formula such that all free variables of ψ occur in α, and $\bar{y}$ is a subtuple of the free variables of α, then $\exists \bar{y}(\alpha \wedge \psi)$ is guarded.

Let us denote the class of queries expressible by guarded formulas by GF, and the class of queries expressible by formulas that are both guarded and connected by CGF. We will establish:

▶ **Theorem 17.** $\mathrm{GF} \cap \mathrm{ADD} = \mathrm{CGF}$.

Towards the proof, we introduce:

▶ **Definition 18.** *The* relaxed version *of CGF, denoted by* CGF^+*, is defined as follows:*

1. *Every atomic formula is in* CGF^+.
2. *If φ and ψ are in* CGF^+*, then so are $\varphi \wedge \psi$, $\varphi \vee \psi$, and $\varphi \wedge \neg\psi$, on condition that the rules for building connected formulas (Section 4.1) are satisfied.*
3. *If α is an atomic formula, and ψ is a boolean combination of* CGF^+ *formulas, all with at least one free variable, and all free variables of ψ occur in α, then both $\alpha \wedge \psi$ and $\exists \bar{y}(\alpha \wedge \psi)$ belong to* CGF^+*, with $\bar{y}$ a subtuple of the free variables of α.*

▶ **Example 19.** The sentence φ: $\exists x, y\,(R(x,y) \wedge (S(x) \vee T(y)))$ belongs to CGF^+, but it is not connected due to the subformula $S(x) \vee T(y)$. However, φ can be equivalently rewritten as $\exists x, y(R(x,y) \wedge S(x)) \vee \exists x, y(R(x,y) \wedge T(y))$ which is in CGF.

In general we note:

▶ **Lemma 20.** *Every* CGF^+ *formula is equivalent to a* CGF *formula.*

We also introduce a subclass of CGF^+ formulas, which we call the *simple guarded formulas*. These are the formulas that can be generated using only syntax rules 1 and 3 of CGF^+ (Definition 18). Simple guarded formulas are useful building blocks. First of all, they are additive. They are q-local if q is their quantifier rank. Furthermore we have the following lemma:

▶ **Lemma 21.** *Every* GF *formula is equivalent to a boolean combination of simple guarded formulas.*

So, we can start the proof of Theorem 17 with an additive boolean combination φ of simple guarded formulas, in disjunctive normal form. Within each clause ψ, we identify the *pseudopositive part* as the part consisting of all positive literals, plus all negative literals with a single free variable. The remainder of the clause is called the *negative part*. We will denote the pseudopositive part of a clause ψ by ψ^{pp} and the negative part by ψ^{neg}.

We may assume that φ has been simplified so that it has no redundancies in the following sense:

(A) If we would remove a subpart of the pseudopositive part of some clause, the resulting formula would not be logically equivalent to φ.

(B) If we would remove an entire clause, the resulting formula would again not be logically equivalent to φ.

(C) No clause contains a negated literal $\neg\eta$ such that η is a sentence that logically implies φ.

If, after these simplifications, we end up with an empty disjunction, then the original φ expresses the n-ary empty query, with n the number of free variables of φ. It is a simple exercise to express this query in CGF.

The plan of the proof is as follows. Each step relies on the additivity of φ; the steps are proven in the given order.

Claim 1: The pseudopositive part of each clause is connected.

Claim 2: In each clause, the free variables of the negative part are included in those of the pseudopositive part.

Claim 3: No clause can have negated literals that are sentences.

Claim 4: All clauses have the same free variables.

The result is a CGF^+ formula and we are done by Lemma 20.

5.2 Complexity of additivity for GF

We show:

▶ **Theorem 22.** *Additivity of guarded formulas is* 2ExpTime*-complete.*

Since satisfiability for guarded formulas is 2ExpTime-hard [9], the hardness follows directly from the simple reduction from satisfiability to additivity given by Proposition 16. Conversely, the membership in 2ExpTime is shown by a reduction from validity to unsatisfiability, which we next describe.

Consider an arbitrary guarded formula φ over a schema $\mathbf{S}$. We will construct a guarded formula φ' over a larger schema $\mathbf{S}^+$ such that φ is additive iff φ' is unsatisfiable. Specifically, $\mathbf{S}^+ = \mathbf{S} \cup \{U_1, U_2\}$, for two fresh unary relation names U_1 and U_2.

Satisfiability for guarded formulas of size n over relations of maximum arity a is decidable in time $2^{O(n)\cdot 2^{a \log a}}$ [10]. If n is the size of φ, our formula φ' will have size $2^{O(n)}$ but the arity does not change. Hence, we will obtain that additivity is in 2ExpTime as desired.

The high-level idea underlying the reduction can be sketched as follows. From φ, we construct guarded formulas φ^+ and $\varphi^\vee$ such that φ is additive iff φ^+ and $\varphi^\vee$ are equivalent over consistent instances (the definitions follow). A crucial property of consistency is that it can be checked via a guarded sentence φ_{cons}. Therefore, φ is additive iff $\varphi_{\mathsf{cons}} \models \varphi^+ \leftrightarrow \varphi^\vee$ iff $\varphi_{\mathsf{cons}} \wedge \neg(\varphi^+ \leftrightarrow \varphi^\vee)$ is unsatisfiable, while $\varphi_{\mathsf{cons}} \wedge \neg(\varphi^+ \leftrightarrow \varphi^\vee)$ is guarded.

The formula φ^+. We inductively define a translation $\cdot^+$ that converts φ into a formula φ^+ over $\mathbf{S}^+$ as follows:

1. If $\varphi = R(x_1, \ldots, x_n)$ for some $R \in \mathbf{S}$, then
$$\varphi^+ \coloneqq (R(x_1, \ldots, x_n) \wedge \bigwedge_{1 \le i \le n} U_1(x_i)) \vee (R(x_1, \ldots, x_n) \wedge \bigwedge_{1 \le i \le n} U_2(x_i)).$$
2. If $\varphi = (x = y)$, then $\varphi^+ \coloneqq (x = y)$.
3. The translation $\cdot^+$ commutes with the boolean connectives, i.e., $(\psi \wedge \chi)^+ \coloneqq \psi^+ \wedge \chi^+$, $(\psi \vee \chi)^+ \coloneqq \psi^+ \vee \chi^+$, and $(\neg\psi)^+ \coloneqq \neg\psi^+$.
4. If $\varphi = \exists y\, \psi$, then $\varphi^+ \coloneqq \exists y\, \psi^+$.

Strictly speaking, the formula φ^+ may not be guarded. However, it is readily transformed into a guarded formula of size $2^{O(|\varphi|)}$, where $|\varphi|$ denotes the size of φ. In what follows, for brevity, we write φ^+ for the exponentially sized rewritten guarded formula.

The formula $\varphi^\vee$. For $k = 1, 2$, we define φ^k as the formula obtained from φ by relativising all quantifiers and free variables to U_k. The latter means that quantifiers are of the form $\exists x \in U_k$, and for every free variable x we have a conjunct $U_k(x)$. A guarded existential formula $\exists y_1, \ldots, y_l(\alpha \wedge \psi)$ can be relativised as $\exists y_1, \ldots, y_l(\alpha \wedge U_k(y_1) \wedge \cdots \wedge U_k(y_l) \wedge \psi)$, which is still guarded. The formula $\varphi^\vee$ is then defined as the (guarded) formula $\varphi^1 \vee \varphi^2$.

Consistency. An $\mathbf{S}^+$-instance $\mathcal{A}^+$ is *consistent* if it admits a partition $(\mathcal{A}_1, \mathcal{A}_2)$ such that:

1. $\mathcal{A}_1$ is an $(\mathbf{S} \cup \{U_1\})$-instance and $\mathcal{A}_2$ is an $(\mathbf{S} \cup \{U_2\})$-instance.
2. $adom(\mathcal{A}_1) \cap adom(\mathcal{A}_2) = \emptyset$.
3. $\{a \mid U_1(a) \in \mathcal{A}_1\} = adom(\mathcal{A}_1)$ and $\{b \mid U_2(b) \in \mathcal{A}_2\} = adom(\mathcal{A}_2)$.

So, consistent instances describe pairs of domain-disjoint instances. Applied to a consistent instance $(\mathcal{A}_1, \mathcal{A}_2)$, we see that φ^+ returns $\varphi(\mathcal{A}_1 \cup \mathcal{A}_2)$, while $\varphi^\vee$ returns $\varphi(\mathcal{A}_1) \cup \varphi(\mathcal{A}_2)$. Hence we obtain:

▶ **Lemma 23.** *The following are equivalent:*

1. *φ is additive.*
2. *For every consistent $\mathbf{S}^+$-instance $\mathcal{A}^+$, $\varphi^+(\mathcal{A}^+) = \varphi^\vee(\mathcal{A}^+)$.*

It remains to show that consistency can be checked via a guarded formula, which will give us the desired reduction. In fact, consistency can be checked via the formula

$$\varphi_{\mathsf{cons}} \coloneqq \neg\exists x(U_1(x) \wedge U_2(x)) \wedge \exists x\, U_1(x) \wedge \exists x\, U_2(x) \wedge \psi_1 \wedge \psi_2 \wedge \chi,$$

where, for $k = 1, 2$, and assuming that $\mathbf{S} = \{R_1, \ldots, R_n\}$ (and R_i is n_i-ary),

$$\psi_k \coloneqq \forall x\,(U_k(x) \to \bigvee_{i=1}^{n} \exists y_1, \ldots, y_{n_i - 1}(R_i(x, y_1, \ldots, y_{n_i-1}) \vee R_i(y_1, x, y_2, \ldots, y_{n_i-1}) \vee \cdots \vee R_i(y_1, \ldots, y_{n_i-1}, x))),$$

and

$$\chi \coloneqq \bigwedge_{i=1}^{n} \forall x_1, \ldots, x_{n_i} \, (R_i(x_1, \ldots, x_{n_i}) \rightarrow ((U_1(x_1) \wedge \cdots \wedge U_1(x_{n_i})) \vee (U_2(x_1) \wedge \cdots \wedge U_2(x_{n_i}))))$$

The above formula can be easily transformed in a guarded formula. We now obtain, as announced, that φ is additive if and only if $\varphi_{\mathsf{cons}} \models \varphi^{+} \leftrightarrow \varphi^{\vee}$.

▶ Remark. At the end of Section 4.4 we gave an example of an FO formula additive over finite instances but not over all instances. Since GF has the finite model property, the above reduction to unsatisfiability implies that a GF formula that is additive over finite instances is automatically additive over all instances.

6 The positive-existential case

In this final section, we concentrate on queries expressed via *positive-existential* (PE) first-order formulas, i.e., formulas that use only $\wedge$, $\vee$, and existential quantification. As always we also use PE to denote the class of all queries expressible in this manner.

It is well-known [1] that PE has the same expressive power as the class of *unions of conjunctive queries (UCQ)*, i.e., disjunctive FO-formulas of the form $\bigvee_{1 \leq i \leq n} \varphi_i(\bar{x})$, with $\varphi_i(\bar{x}) = \exists \bar{y} \, (\alpha_{i,1} \wedge \cdots \wedge \alpha_{i,m_i})$, where each $\alpha_{i,j}$ is an atomic formula. Formulas of the form $\varphi_i(\bar{x})$ are called *conjunctive queries*. In earlier work [6], two of us already showed that UCQ $\cap$ ADD = CUCQ, where CUCQ refers to the connected UCQs. As a consequence, also PE $\cap$ ADD = CPE, where CPE refers to the connected PE formulas.

In this section, we focus on the complexity of deciding additivity for PE formulas. For UCQs, additivity was already shown to be NP-complete [6]. Unfortunately, in general, it takes exponential time to convert a PE formula into a union of conjunctive queries. Thus, the naive algorithm provides only an exponential time upper bound, which is not optimal. We can, however, show that our problem lies at the second level of the polynomial hierarchy:

▶ **Theorem 24.** *Additivity of* PE *formulas is* Π_2^P*-complete. The lower bound holds even over unary and binary relations.*

Towards the proof, let us first introduce some useful notions. A *component* of a conjunctive query $\varphi_i(\bar{x}) = \exists \bar{y} \, (\alpha_1 \wedge \cdots \wedge \alpha_n)$ is a connected formula $\psi = \exists \bar{y} \, (\alpha_{i_1} \wedge \cdots \wedge \alpha_{i_m})$, where $\{i_1, \ldots, i_m\} \subseteq \{1, \ldots, n\}$, such that, for every $j \in \{1, \ldots, n\} \setminus \{i_1, \ldots, i_m\}$, the formula $\alpha_{i_1} \wedge \cdots \wedge \alpha_{i_m} \wedge \alpha_j$ is not connected anymore. Given two formulas $\varphi(\bar{x})$ and $\psi(\bar{x})$ over a schema $\mathbf{S}$, we say that φ is *contained* in ψ, written $\varphi \subseteq \psi$, if $\varphi(\mathcal{A}) \subseteq \psi(\mathcal{A})$ for every $\mathbf{S}$-instance $\mathcal{A}$. We need the following:

▶ **Lemma 25** ([6])**.** *Consider a union of conjunctive queries $\varphi(\bar{x})$ of the form $\bigvee_{1 \leq i \leq n} \varphi_i(\bar{x})$. The following are equivalent:*

- *φ is additive;*
- *for each $i \in \{1, \ldots, n\}$, there exists a component $\psi(\bar{x})$ of $\varphi_i(\bar{x})$ such that $\psi \subseteq \varphi$.*

Consider an arbitrary PE-formula φ. Even though we cannot efficiently convert φ into an equivalent union of conjunctive queries $\bigvee_{1 \leq i \leq n} \varphi_i$, we can non-deterministically construct a disjunct φ_i in polynomial time. This fact, together with Lemma 25, leads to the following non-deterministic algorithm for checking whether φ is *not* additive:

1. Non-deterministically construct a disjunct φ_i of the union of conjunctive queries obtained after converting φ into an equivalent union of conjunctive queries.
2. Compute the set C of all the components of φ_i.
3. If, for each $\psi \in C$, it holds that $\psi \not\subseteq \varphi$, then ACCEPT; otherwise, REJECT.

It is easy to verify, due to Lemma 25, that φ is *not* additive iff the above procedure accepts. Observe now that steps 1 and 2 are feasible in polynomial time. Finally, during step 3, we need to check polynomially many times for non-containment of a conjunction of atomic formulas, i.e., a conjunctive query, into a PE-formula. The latter is feasible in coNP. Thus, the obtained upper bound for non-additivity is $\mathrm{NP}^{\mathrm{NP}} = \Sigma_2^P$, as needed.

It remains to establish that additivity for PE-formulas is Π_2^P-hard. This is shown by a reduction from the problem of containment for PE-sentences, i.e., given two PE-sentences φ and ψ, to decide whether $\varphi \subseteq \psi$, which is Π_2^P-hard [19]. Actually, this problem remains Π_2^P-hard even if the left-hand side formula is connected, and the relation symbols have arity at most two; this is implicit from the work by Bienvenu et al. [7].

Given a connected PE-formula φ, and an arbitrary PE-formula ψ, we can show that $\varphi \subseteq \psi$ iff the PE-formula $\varphi \wedge P(x_\varphi) \wedge \psi$ is additive, where x_φ is an existentially quantified variable in φ, and P a fresh relation name. This shows that additivity for PE-formulas is Π_2^P-hard even for unary and binary relations.

7 Conclusion

We conclude with a few directions for further research.

1. Investigate the complexity, in terms of formula length, of the shortest connected formula equivalent to an additive formula.
2. Our reduction from additivity to unsatisfiability for GF formulas is exponential. However, the GF formula that is produced has exponential length only to get rid of disjunctive guards. If disjunctive guards would be allowed, the reduction would be polynomial and would preserve bounded arity. We conjecture that additivity for GF formulas of bounded arity is actually in ExpTime, and plan to investigate this in the near future.
3. Also, our reduction from additivity to unsatisfiability works in general, for FO formulas, and for other reasonable logics. It is interesting to apply the reduction to other decidable logics, and see if the reduction produces a formula in the same logic. For example, our reduction also works for FO^2, thus additivity for FO^2 formulas is decidable.
4. Give a short classical model-theoretic proof, e.g., involving the compactness theorem, that every FO formula, additive over all structures, is equivalent to a CFO formula.
5. Our version for additivity is rather strict in the context of distributed computation. Considering classes of queries that distribute over other kinds of partitions for instances is an interesting line for future research.

References

1 S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

2 N. Alechina and Y. Gurevich. Syntax vs semantics on finite structures. In J. Mychielski, G. Rozenberg, and A. Salomaa, editors, *Structures in Logic and Computer Science*, volume 1261 of *Lecture Notes in Computer Science*, pages 14–33. Springer, 1997.

3 T.J. Ameloot, B. Ketsman, F. Neven, and D. Zinn. Weaker forms of monotonicity for declarative networking: A more fine-grained answer to the CALM-conjecture. *ACM Transactions on Database Systems*, 40(4):article 21, 2016.

4 T.J. Ameloot, B. Ketsman, F. Neven, and D. Zinn. Datalog queries distributing over components. *ACM Transactions on Computational Logic*, 18:article 5, 2017.

5 H. Andréka, I. Németi, and J. van Benthem. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998.

6 G. Berger and A. Pieris. Ontology-mediated queries distributing over components. In S. Kambhampati, editor, *Proceedings 25th International Joint Conference on Artificial Intelligence*, pages 943–949. IJCAI/AAAI Press, 2016.

7 M. Bienvenu, C. Lutz, and F. Wolter. Query containment in description logics reconsidered. In G. Brewka, T. Eiter, and S.A. McIlraith, editors, *Principles of Knowledge Representation and Reasoning: Proceedings 13th KR*. AAAI Press, 2012.

8 H. Gaifman. On local and nonlocal properties. In *Proceedings of the Herbrand symposium (Marseilles, 1981)*, volume 107 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1982.

9 E. Grädel. On the restraining power of guards. *Journal of Symbolic Logic*, 64(4):1719–1742, 1999.

10 E. Grädel and I. Walukiewicz. Guarded fixed point logic. In *Proceedings 14th Annual IEEE Symposium on Logic in Computer Science*, pages 45–54, 1999.

11 J.M. Hellerstein. The declarative imperative: Experiences and conjectures in distributed logic. *SIGMOD Record*, 39(1):5–19, 2010.

12 D. Leinders, M. Marx, J. Tyszkiewicz, and J. Van den Bussche. The semijoin algebra and the guarded fragment. *Journal of Logic, Language and Information*, 14:331–343, 2005.

13 D. Leinders and J. Van den Bussche. On the complexity of division and set joins in the relational algebra. *J. Comput. Syst. Sci.*, 73(4):538–549, 2007.

14 M. Otto. Elementary proof of the van Benthem-Rosen characterization theorem. Fachbereich Informatik online preprint 2342, TU Darmstadt, 2004. URL: `http://www3.mathematik.tu-darmstadt.de/fb/mathe/preprints.html`.

15 M. Otto. Modal and guarded characterisation theorems over finite transition systems. *Annals of Pure and Applied Logic*, 130:173–205, 2004.

16 M. Otto. Bisimulation invariance and finite structures. In Z. Chatzidakis, P. Koepke, and W. Pohlers, editors, *Logic Colloquium '02*, volume 27 of *Lecture Notes in Logic*, pages 276–298. Cambridge University Press, 2006.

17 E. Rosen. Modal logic over finite structures. *Journal of Logic, Language and Information*, 5:427–439, 1997.

18 E. Rosen. Some aspects of model theory and finite structures. *Bulletin of Symbolic Logic*, 8:380–403, 2002.

19 Y. Sagiv and M. Yannakakis. Equivalence among Relational Expressions with the Union and Difference Operators. *J. ACM*, 27(4):633–655, 1980.

20 D. Surinx, J. Van den Bussche, and D. Van Gucht. The primitivity of operators in the algebra of binary relations under conjunctions of containments. In *Proceedings 32nd Annual ACM/IEEE Symposium on Logic in Computer Science*. IEEE Computer Society Press, 2017.

21 D. Surinx, J. Van den Bussche, and D. Van Gucht. A framework for comparing query languages in their ability to express boolean queries. In *Foundations of Information and Knowledge Systems*, pages 360–378, 2018.

22 J.D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume I. Computer Science Press, 1988.

23 J. van Benthem. *Modal Logic and Classical Logic*. Bibliopolis, Naples, 1983.